LLNL-TR-421078

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Advances in Bayesian Model Based Clustering Using Particle Learning

D. M. Merl

December 7, 2009

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Advances in Bayesian Model Based Clustering Using Particle Learning

BY DANIEL MERL
Systems and Intelligence Analysis
Lawrence Livermore National Laboratory

November 19, 2009

1 Introduction

Recent work by Carvalho, Johannes, Lopes and Polson [1] and Carvalho, Lopes, Polson and Taddy [2] introduced a sequential Monte Carlo (SMC) alternative to traditional iterative Monte Carlo strategies (e.g. MCMC and EM) for Bayesian inference for a large class of dynamic models. The basis of SMC techniques involves representing the underlying inference problem as one of state space estimation, thus giving way to inference via particle filtering. The key insight of Carvalho et al was to construct the sequence of filtering distributions so as to make use of the posterior predictive distribution of the observable, a distribution usually only accessible in certain Bayesian settings. Access to this distribution allows a reversal of the usual propagate and resample steps characteristic of many SMC methods, thereby alleviating to a large extent many problems associated with particle degeneration. Furthermore, Carvalho et al point out that for many conjugate models the posterior distribution of the static variables can be parametrized in terms of [recursively defined] sufficient statistics of the previously observed data. For models where such sufficient statistics exist, *particle learning* as it is being called, is especially well suited for the analysis of streaming data do to the relative invariance of its algorithmic complexity with the number of data observations [1]. Through a particle learning approach, a statistical model can be fit to data as the data is arriving, allowing at any instant during the observation process direct quantification of uncertainty surrounding underlying model parameters.

Here we describe the use of a particle learning approach for fitting a standard Bayesian semiparametric mixture model as described in Carvalho, Lopes, Polson and Taddy [2]. In Section 2 we briefly review the previously presented particle learning algorithm for the case of a Dirichlet process mixture of multivariate normals. In Section 3 we describe several novel extensions to the original implementation of Carvalho et al that allow us to retain the computational advantages of

[1]By this I mean the amount of processing per observation does not depend on the total number of observations, whereas a single iteration of MCMC for a semiparametric model often scales in proportion to the total number of observations

particle learning while improving the suitability of the methodology to the analysis of streaming data and simultaneously facilitating the real time discovery of latent cluster structures. Section 4 demonstrates our methodological enhancements in the context of several simulated and classical data sets, showcasing the use of particle learning methods for online anomaly detection, label generation, drift detection, and semi-supervised classification, none of which would be achievable through a standard MCMC approach. Section 5 concludes with a discussion of future directions for research.

2 Particle Learning for Dirichlet Process Mixtures

Dirichlet process mixture (DPM) models are appealing for the fact that they do not require the specification *a priori* of the number of components in the mixture. This makes the DPM the current *de facto* choice for flexible density estimation.

2.1 Dirichlet Process Mixtures

The most common nonparametric model in the Bayesian literature is by far the Dirichlet process mixture of multivariate normals. For an observation x_t, the model is specified as follows:

$$\begin{aligned} x_t|\mu_t,\Sigma_t &\sim N(\mu_t,\Sigma_t) && (1)\\ (\mu_t,\Sigma_t)|G &\sim G && (2)\\ G|\alpha_0,G_0 &\sim DP(\alpha_o G_0) && (3)\\ \alpha_o &\sim Gamma(e,f) && (4)\end{aligned}$$

where the Dirichlet process base measure G_0 is usually the conjugate Normal Inverse Wishart prior for (μ_t,Σ_t). For convenience we will use θ_t to denote (μ_t,Σ_t). Early foundational work by Ferguson (1974) showed that the marginal distribution of θ_t given the preceding $\theta_1\ldots\theta_{t-1}$ has a Polya Urn / Chinese Restaurant Process representation,

$$\theta_t|\{\theta^*_j\}_{j=1}^{n^*},\{n_j\}_{j=1}^{n^*},\alpha_0,G_0 \sim \frac{\alpha_0}{\alpha_0+t-1}G_0+\frac{1}{\alpha_0+t-1}\sum_{j=1}^{n^*}n_j\delta_{\theta^*_j} \quad (5)$$

where $\{\theta^*_j\}_{j=1}^{n^*}$ denote the unique component specific parameters from amongst $\theta_1..\theta_{t-1}$ and $\{n_j\}_{j=1}^{n^*}$ indicate the number of preceding observations associated with the given component (introducing the previously mentioned latent configuration variable k_i for each observation i, we have $n_j=\sum_{i\in 1:t-1}1_{k_i=j}$). The Dirichlet process is therefore well suited for use as a prior distribution on the mixing distribution of the mixture model, guaranteeing an unknown but countable number of mixture components.

Letting $x_1,x_2,\ldots,x_{t-1}$ denote an ordered sequence of observations, we will be interested in the posterior predictive distribution of observation x_t as a function of the already observed

$x_1, x_2, \ldots x_{t-1}$, specifically, a function of the data maps the already observed data to a set of sufficient statistics for each θ_j^*. Since each component in the mixture model is a multivariate normal kernel, the set $s_j = \{\sum_{i:k_i=j} x_i, \sum_{i:k_i=j} x_i' x_i, n_j\}$ provides sufficient statistics for each θ_j^*. Thus given $\{s_j\}_{j=1}^{n^*}$ we have

$$p(x_t|\{s_j\}_{j=1}^{n^*}, \alpha_0, G_0) = \int\int \ldots \int N(x_t|\theta_t) p(\theta_t|\{\theta_j^*\}, \{n_j\}, \alpha_0, G_0) \times \prod_{j=1}^{n^*} p(\theta_j^*|s_j, \alpha_0 G_0) \mathrm{d}\theta_t \mathrm{d}\theta_1^* \ldots \mathrm{d}\theta_{n^*}^* \quad (6)$$

This cumbersome integral simply averages over the uncertainty surrounding (1) from which mixture component the the new x_t is to be drawn (viz., the Polya urn structure of the random distribution), and (2) the location and shape of that mixture component based on the current normal inverse Wishart posterior involving the sufficient statistics for the component. Letting $G_0(\theta) = NIW_\nu(\theta|\mu_0, \gamma, \Sigma_0)$ (with $E(\Sigma) = \Sigma_0/(\nu-2)$), the expression simplifies to

$$p(x_t|\{s_j\}_{j=1}^{n^*}, \alpha_0, G_0) = \int N(x_t|\theta) \left\{ \frac{\alpha_0}{\alpha_0 + t - 1} NIW_\nu(\theta|\mu_0, \gamma, \Sigma_0) + \sum_{j=1}^{n^*} \frac{n_j}{\alpha_0 + t - 1} NIW_{\nu+n_j}\left(\theta|\mu_j, \frac{\gamma}{1+\gamma n_j}, \Sigma_j\right) \right\} \mathrm{d}\theta \quad (7)$$

or

$$= \frac{\alpha_0}{\alpha_0 + t - 1} St_\nu \left(x_t|\mu_0, \frac{1+\gamma}{\nu}\Sigma_0 \right) + \sum_{j=1}^{n^*} \frac{n_j}{\alpha_0 + t - 1} St_{\nu+n_j}\left(x_t|\mu_j, \frac{1+\gamma n_j + \gamma}{(1+\gamma n_j)(\nu+n_j)} \Sigma_j \right) \quad (8)$$

where $\mu_j = \frac{\mu_0 + \gamma n_j \bar{x}_j}{1+\gamma n_j}$ and $\Sigma_j = \Sigma_0 + \sum_{i:k_i=j}(x_i - \bar{x}_j)'(x_i - \bar{x}_j) + \frac{n_j}{1+\gamma n_j}(\bar{x}_j - \mu_0)'(\bar{x}_j - \mu_0)$. Note that both of these expressions involve terms that can be computed from the sufficient statistics s_j. The key idea behind particle learning is now to regard the remaining uncertainty surrounding the mapping of data to sufficient statistics as a state space estimation problem, to be handled via filtering in the following way.

2.2 The PL Algorithm

Carvalho et al described a particle learning algorithm for a Dirichlet process mixture model as follows.

1. Initialize a particle cloud based on observation x_0. Each identical particle consists of the state $z_0 = \{\alpha_0, \mu_0, \Sigma_0, s_0 = (x_0, x_0' x_0, 1)\}$, where α_0,μ_0,and Σ_0 are assigned some initial value.

2. After observing x_t, *resample* with replacement from among the current particles with weights proportional to Equation 8.

3. For the newly sampled set of particles, *propagate* each particle by first assigning observation x_t to one of the existing mixture components with probability proportional to

$$\frac{n_j}{\alpha_0 + t - 1} St_{\nu + n_j}\left(x_t | \mu_j, \frac{1 + \gamma n_j + \gamma}{(1 + \gamma n_j)(\nu + n_j)}\Sigma_j\right)$$

   or to a new mixture component with probability proportional to

$$\frac{\alpha_0}{\alpha_0 + t - 1} St_{\nu}\left(x_t | \mu_0, \frac{1 + \gamma}{\nu}\Sigma_0\right).$$

   This amounts to the standard Gibbs step for sampling the configuration variable k_t. Having sampled this configuration variable for the particle, update the particle's sufficient statistics as appropriate.

4. For each particle, perform standard Gibbs updates for α_0, μ_0, Σ_0 (see Appendix).

This can be interpreted in terms of sequential importance sampling with resampling. The aim of any filtering algorithm is to provide successive, approximate draws from the current posterior distribution of the hidden state variable (z_t). Suppose then that we have a Monte Carlo approximation to $p(z_{t-1}|x^{t-1})$. Upon observation of x_t, our goal is to produce samples from $p(z_t|x^t)$ (the superscript denotes all observations up to and including time t). First, we rewrite $p(z_t|x^t)$ as

$$p(z_t|x^t) = \int p(z_t|z_{t-1}, x^t)p(z_{t-1}|x^t)\mathrm{d}z_{t-1}.$$

We can obtain a Monte Carlo approximation to this integral if (1) we are able to obtain samples from $p(z_{t-1}|x^t)$; and (2) we are able to evaluate $p(z_t|z_{t-1}, x^t)$. Regarding (1), note that

$$\begin{aligned}
p(z_{t-1}|x^t) &= p(z_{t-1}|x_t, x^{t-1}) \\
&= \frac{p(x_t|z_{t-1}, x^{t-1})p(z_{t-1}|x^{t-1})}{p(x^t|x^{t-1})} \text{ (by Bayes rule)} \\
&\propto p(x_t|z_{t-1})p(z_{t-1}|x^{t-1}) \text{ (since } p(x_t|z_{t-1}, x^{t-1}) \text{ does not depend on } x^{t-1})
\end{aligned}$$

Thus if samples from $p(z_{t-1}|x^{t-1})$ are available, samples from $p(z_{t-1}|x^t)$ can be obtained by importance sampling, using a weight function equal to the predictive distribution $p(x_t|z_{t-1})$. In the setting of Dirichlet process mixtures, this predictive distribution is simply that described by Equation 8. After performing this resampling step, each particle represents a uniformly weighted draw from $p(z_{t-1}|x^t)$. Now regarding (2), the distribution $p(z_t|z_{t-1}, x^t)$ is simply the current posterior distribution of the hidden state. This is evident when the distribution is rewritten as $p(z_t|z_{t-1}, x^t) \propto p(x_t|z_t, z_{t-1})p(z_t|z_{t-1}) = p(x_t|z_t)p(z_t)$ (conditional on z_t, the distributions do not depend on z_{t-1}). Here $p(z_t)$ denotes the joint prior distributions on the various elements of the hidden state vector z_t. Therefore, therefore, using the Monte Carlo samples from $p(z_{t-1}|x^t)$, we have the following expression for the Monte Carlo estimate of $p(z_t|x^t)$:

$$\begin{aligned} E^{MC}_{z_{t-1}}(p(z_t|x^t)) &= \sum_i p(z_t|z^{(i)}_{t-1}, x^t)w(z^{(i)}_{t-1} \\ &= \frac{1}{N} p(z_t|z^{(i)}_{t-1}, x^t) \end{aligned}$$

Thus, to sample from this distribution, we simply sample from the current posterior distributions implied by each particle. For the class of models considered here, this involves the allocation of observation x_t to one of the existing mixture components, and the α_0, μ_0 and Σ_o parameters associated with the Dirichlet process.

3 Advantages of Inference via Particle Learning

Particle learning affords several major advantages over MCMC-based inference for certain domains of application, especially those pertaining to the analysis of streaming data and the use of the DPM as a clustering tool. Here we describe these advantages along with several computational enhancements that improve upon the basic algorithm.

3.1 Data Stream Representations

Since particle learning, unlike MCMC, does not assume a fully observed data set, and since the previously observed data impacts the model fit only through the sufficient statistics, it is not strictly necessary to retain an observation after the associated resample and propagate steps have been completed. In terms of implementation and application this means that that data object need not be a static file representing observations over a finite time interval, as is usually the case in applied statistics. Rather, the data can be represented as a stream directly (as by an open connection to a port), and the duration of the analysis can be unknown and/or indefinite. Note however that for indefinite observation, measure must be taken to prevent numerical overflow that can occur as t grows arbitrarily large. This is discussed below in the section on drift detection.

3.2 Automatic Anomaly Detection

The key advantage of a sequential approach is obviously its amenability to sequential data. Whereas MCMC and EM-based methods are only able to produce uncertainty statements regarding the data as observed in its entirety, the particle learning approach is unique in its ability to express the state of uncertainty at a particular instant in the observation process. This makes particle learning ideal for the analysis of streaming data for which the goal is the *online* discovery of the structure and trends characterizing the data, and deviations thereof. For example, suppose we are observing an incoming data stream and want to be alerted when an anomalous observation arrives. A direct measure of the degree of anomaly of a given observation is the proportion of particles in the current cloud for which that observation caused the creation of a new mixture component during the propagate step. For each particle, recall that the probability associated with adding a new mixture component to accommodate x_t is

$$\pi_t^{(i)} \quad \propto \quad \left(\frac{\alpha_0^{(i)}}{\alpha_0^{(i)} + t - 1} St_\nu \left(x_t | \mu_0^{(i)}, \frac{1+\gamma}{\nu} \Sigma_0^{(i)} \right) \right)$$

Then $\pi_t = \frac{1}{N} \sum_i \pi_t^{(i)}$ is the Monte Carlo estimate of this probability of anomaly, averaged over the uncertainty surrounding the underlying hidden states. Note that this measure reflects the degree of anomaly of an observation only at the time of its arrival. It may be the case that an anomaly, by this measure, simply marks the first occurrence of what becomes a heavily weighted mixture component. However that recognition is inherently *retrospective*, and at the time of arrival all that can be said is the degree to which the observation is sufficiently distinct from all that preceded it. First arrivals can be separated from true anomalies later in the observation process by inspection of the persistence and weight of the components founded by the observation.

3.3 Nonstationary Mixtures and Drift Detection

A natural generalization of the stationary mixture model is the nonstationary mixture model, in which the locations and shapes of the mixture components are allowed to change over time. For example, suppose that the observed data stream represents a mixture of an unknown number of separate sub-streams (for example, suppose the data arises from a mixture of autoregressive processes). It might be seen as something of an abuse of mixture modeling techniques to employ them in such settings rather than to directly model the data as a mixture of, say, dynamic linear models [13]. Such an approach is worth investigating. However, the standard mixture model framework can be adapted to the nonstationary setting so easily that it should serve well as a first approximation to a mixture of time series model.

The strategy is simply to introduce a decay factor λ to be applied when updating the sufficient statistics of a component. For example, if during a propagate step an observation x_t is assigned to a mixture component j, the vector s_j of sufficient statistics for component j is updated as follows:

$$s_j \quad = \quad [\lambda n_j + 1 \, , \, \lambda \sum x_j + x_t \, , \, \lambda \sum x_j' x_j + x_t' x_t]$$

For all other components, the vector of sufficient statistics is degraded as:

$$s_{j'} \quad = \quad [\lambda n_{j'} \, , \, \lambda \sum x_{j'} \, , \, \lambda \sum x_{j'}' x_{j'}]$$

In these expressions, the second and third entries in the vector of sufficient statistics are the sum of x and sum of x-transpose-x for observations assigned to the component.

The effect of this decay factor is to force the mixture model to weight more heavily recent observations. In this way, nonstationary mixture distributions can be emulated. This capability will be especially important for facilitating *drift detection*, as well preventing numerical overflow errors associated with maintaining sufficient statistics for arbitrarily long periods of observation. Most families of ensemble classifiers assume that the distribution of features that distinguish one subtype from another are static. If the underlying distributions of these features are in fact nonstationary, eventually the classifier's training will become irrelevant. Moreover, such classifiers have no mechanism for detecting when those underlying distributions have drifted sufficiently that retraining is necessary. One intended application of the methodology described here is to facilitate retraining schedules for other classifiers by detecting substantial changes to the underlying mixture distribution, and producing new labels for unlabeled data which can be provided to other classifiers for retraining purposes, or possibly used in place of the now obsolete classifier.

3.4 Window-based Particle Rejuvenation

Although the basic particle learning algorithm does not require the retention of past observations, it can be useful to *rejuvenate* resampled particles periodically. In the sequential Monte Carlo literature, rejuvenation usually refers to the mechanisms by which particle degeneracy (i.e. the loss of heterogeneity of the cloud) is prevented. For the particle-learned DPM, we may wish to perform some additional MCMC steps to reassign of observations amongst mixture components, or to split or merge existing mixture components in a particle. To do so requires the values of those past observations, and the configuration variables indicating to which component the observation is currently associated. Rather than retain all past observations and configuration variables, we have implemented a sliding window scheme whereby the most recent N observation and configuration variables are retained. The size of the window can be specified at run time, and adjusted according to computational/memory limitations. By retaining only a fixed number of observations at a time, we are able to balance the utility of particle rejuvenation without requiring a significant augmentation of the state space represented by each particle. In the context of nonstationary mixtures, the decay factor provides a natural guide for choosing the window width, given that the effective number of observations represented by the mixture model will be $1/(1-\lambda)$. As will be demonstrated below, split/merge steps, such as those described in Jain and Neal [9] and Dahl [3] will be especially useful in the nonstationary setting (see Appendix).

3.5 A Solution to the Label Switching Problem

Although originally intended for density estimation, Bayesian semiparametric mixture models, and in particular the Dirichlet process mixture model, have gained much popularity in recent years as methods for performing model-based clustering. Inference on mixture models is usually accomplished by augmenting the parameter space with a set of latent indicator variables by which an observation is associated with a single component in the mixture. These indicator variables can be regarded to indicate cluster membership. It might be hoped that by conducting inference on these indicator variables using standard Bayesian technologies such as MCMC we may able to quantify uncertainty surrounding the underlying cluster structure. However, there are significant technical challenges to using these indicator variables to make *probabilistic* statements about cluster membership. As discussed in Merl and West [11], these challenges arise from what is known as the label switching problem (see also Stephens [12]). Label switching arises from the invariance of the model likelihood to permutations of the order of mixture components. In practical terms, label switching precludes statements such as "*observation i has probability of being in cluster j*" because the term "*cluster j*" has no persistent meaning over the course of the MCMC.

In the PL setting, it becomes possible to ascribe labels to mixture components because the components are now uniquely identified even while the location and shape of the component remains uncertain. One such label is the index of the observation that caused the creation of the component. As successive observations arrive and particles are propagated, the location and shape of each component founded by a particular observation will differ from particle to particle, however the interpretation of the component will be retained. This labeling strategy is facilitated by the sliding window rejuvenation scheme described above, which serves to increase the potential for consensus across particles regarding the first observation to be associated with a new component. We can then evaluate the probability that an observation will receive one of the current labels by averaging the particle-specific label probabilities. This will be described below.

3.6 Adaptive Classifiers

Traditionally, model based clustering has focused solely on the problem of generating labels for unlabeled data, and interpretation of cluster membership has remained a task for the practitioner. This is at odds with the *supervised learning* methods of the machine learning literature, in which labeled data is used to train a classifier, which is then validated by its prediction accuracy. A fundamental weakness of such classifiers is their inability to adaptively update their prediction rules, as to re-characterize the rule associated with a particular known type, or to add a new rule for a previously unseen type, without significant retraining. Meanwhile, a fundamental weakness of model based clustering has been the inability to ascribe cluster labels of any type, let alone cluster labels representing both known and unknown types.

Having now apparently resolved the labeling problem, we can begin to devise training schemes using PL. We can think of this training process as a way to generate an informed prior distribution for our mixture model, or more specifically, an informed particle cloud from which starting point subsequent analyses can be initialized. One possible procedure is as follows.

1. Given a training set $\{x\}^{t_1}, \{x\}^{t_2}, \ldots, \{x\}^{t_J}$, where each $\{x\}^{t_i}$ is a set of observations associated with a known type t_i, fit a separate DPM to each collection of $\{x\}^{t_i}$.

2. Construct a new particle cloud in which each particle p contains a collection of components culled from the preceding step as follows:

   (a) For each known type t_i

   (b) For each label ℓ generated during the analysis of $\{x\}^{t_i}$

(c) Randomly choose a component j with label ℓ from amongst all particles in this analysis containing a component with label ℓ.

(d) "Inject" $\{s_j\}$ (the sufficient statistics associated with that component) into p, but relabel this component t_i.

3. Begin PL analysis of unlabeled data $\{y\}$.

Note that under this approach, by injecting each new particle with a randomly sampled component for each label, we are able to retain uncertainty regarding the location and shape of components associated with the labels generated for each known type.

As analysis proceeds, for each observation y_t with unknown type, we can calculate the probability of y_t associating with an component labeled with one of the known types, while allowing for the possibility that y_t represents the first arrival of a previously unseen type. Over the course of analysis, the characterization of the distributions associated with the known types will be updated in light of new observations, and new labels will be generated to characterize new types. In this way, PL can serve as the basis for an adaptive classifier, combining the benefits of a formal statistical model with the interpretability of supervised classifier.

3.7 Parallelization

It is worth mentioning that even more so than MCMC, particle methods are trivially parallelized. In an optimal scenario[2], we would have a 1:1 ratio of particles to processing units. After the arrival of a new observation, the value of that observation is distributed to each particle so that the resampling weight of the particle can be calculated. All these calculations are independent, and can be performed in parallel. Though a central thread is required to collect the weights and perform the resampling, the process of repopulating the processing units with resampled particle is efficient since the state space of each particle is bounded and generally relatively small. All propagate steps can then be performed in parallel as well. Thus given a sufficient number of processing units, it will be possible to reduce between-observation latency to the point that real time analysis can be achieved. Even given a modest number of processing units, desired latencies can be achieved by combining parallel computing with approximate inference strategies as described in Appendix C.

We are also currently investigating opportunities for exploiting graphics processing units (GPUs) to further enhance within particle computation, the matrix operations associated with which would be especially well suited to the strengths of modern GPUs. Several recent technical reports have indicated that such an approach could confer significant advantages to filtering-based algorithms (see [7] and [10]).

[2]Technically there is a "more optimal" scenario in which each particle is associated with n^* processing units such that even its own internal weight and propagate calculations could be parallelized.

4 Simulations

4.1 1-D Simulation

Figure 1 shows the fitted density obtained by performing PL-based inference on data simulated from a simple 3-component mixture of normals. The estimated density function shown is eval-

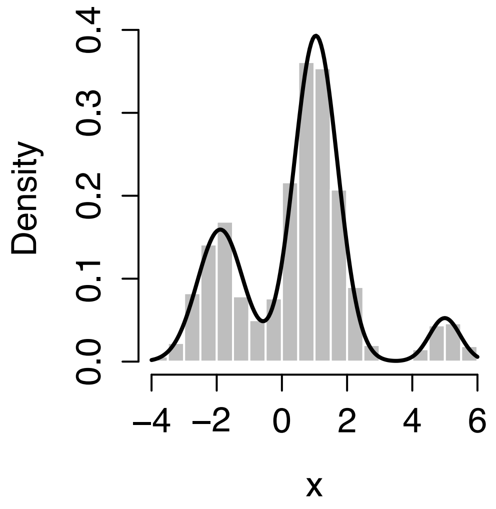


Figure 1: Histogram of simulated data, overlayed with fitted density.

uated using the final state of the particle cloud:

$$p(x^{new}|x^T) \approx \frac{1}{N}\sum_{i=1}^{N} p(x^{new}|\{s_j^{(i)}\}_{j=1}^{n^*_{(i)}}, \alpha_0^{(i)}, G_0^{(i)})$$

Here the (i) subscript or superscript indicates the value associated with particle i (each particle has its own set of component sufficient statistics s_j, DP scale parameter α_0, and DP base measure G_0). The goodness-of-fit evidenced here is standard for DP mixtures.

Although the exchangeability of observations is a fundamental assumption of most mixture models, here we have assumed that there is an important structure to the sequence of arrivals. The simulated observations are sorted such that initially, all observations are generated from the same component. Around time 300, observations begin to arrive that were generated from the second component, and around time 1000 observations begin to arrive from the third component. At this point, the exchangeability assumption still holds in that the fitted mixture model is invariant to reorderings of the sequence of observations, but the labeling and anomaly detection schemes described above will exploit the observed ordering. Figure 2 depicts this arrival pattern, along with the probabilities of anomaly for each observation. The probability of anomaly is

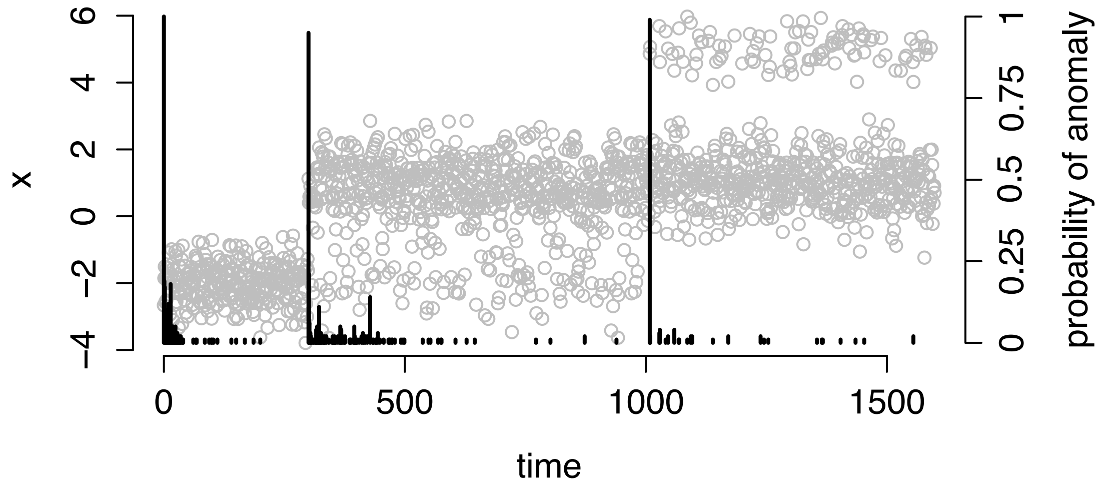


Figure 2: Simulated stationary data streams, with probability of anomaly for each observation.

naturally defined by the proportion of particles in which the observation causes a new mixture component to be created. These probabilities show spikes at observations 0, 300, and 1008 (the spike at observation 0 simply indicates that the particle cloud was initialized with all particles containing a single component labeled 0), and thus clearly delineate the first arrivals of observations from a yet-unseen mixture component.

4.2 Nonstationary Mixtures

The next several simulations demonstrate the ability of the method to accommodate the situation of nonstationary mixtures through the use of the decay factor and split/merge particle rejuvenation. For each simulation we show first the results obtained by application of the standard stationary mixture model (i.e. $\lambda = 1$), and then the results associated with $\lambda = 0.99$, with a single split/merge step following each propagate step. The split/merge procedure employed here is a Metropolis-Hastings step, as described by Dahl [3]. Briefly, this procedure is as follows. Two observations are sampled uniformly without replacement from among the most recent $1/(1-\lambda)$ observations. Then, for each particle, if those observations are currently assigned to different mixture components, it is proposed to merge the two components into a single component. If the observations are assigned to different mixture components, it is proposed to split the single component into two components, with the new components *founded by* one of the chosen observations. All other observations currently assigned to the single component are then assigned to one of the two new components following the usual propagate step distributions. For each particle, the Metropolis-Hastings acceptance probability for the move is evaluated, and the state of the particle is updated appropriately. By choosing the two observations at a global level and then applying the split/merge procedure to all particles using the same observations, we are able to retain the labeling scheme previously described. In the case of a merge, the label of the merged-component is defined to be the label of the earlier of the two parent components. In the case of a split, the label of the smaller component (in terms of number of observations assigned to it) is defined to be the index of the randomly chosen observation that founded the new component. The label of the larger component is defined to be the same as the label of the parent component. In this way, when a split or merge occurs, there will be consensus amongst the particles as to the label of the new component. For more details, refer to the Appendix.

4.2.1 Drifting Data Streams

Figure 3 depicts the simplest type of nonstationary mixture, in which data arises from a single normal component whose mean is changing over time. Essentially, this data might be exactly modeled by a standard regression with Gaussian noise. In each panel, the data is color-coded by its maximum probability label as determined at the time of its arrival, and the predictive distribution of the data is plotted vertically at regular time intervals. The top panel depicts the results of fitting a standard, stationary mixture ($\lambda = 1$) to the drifting data. As the data continues to drift, the mixture model framework creates additional components to represent the increasing spread of the data. The clustering induced by the mixture components in this setting is clearly misleading, especially compared to the clustering induced under the nonstationary setting ($\lambda = 0.99$). With $\lambda = 0.99$, the single mixture component is able to track the drifting data without the introduction of spurious components, and without monotonically increasing the variance of the predictive distribution.

4.2.2 Splitting Data Streams

The next simulation (Figure 4) depicts a situation in which a single data stream splits into two [drifting] data streams. In the top panel, under the assumption of a stationary underlying mixture, the diverging data streams result in a single, high variance component accounting for all observations. This situation may be seen as a sort of pathological case for the standard mixture model. The initial sequence of observation (prior to time 300) are adequately and accurately modeled by a single component. As the divergence begins, the diverging observation continue to be assigned to the single component, which currently has most of its mass around $x = 0$. The continued assignment of the diverging observations to this component results in the strangely peaked, high variance predictive distribution depicted at $t = 749$. In contrast, by using a decay factor of $\lambda = 0.99$ and a single split-merge proposal per propagate step, the nonstationary model (lower panel) is able to recognize the divergence by creating two components that independently track the continued drift of the new streams without sacrificing predictive variance.

4.2.3 Merging Data Streams

The final simulation (Figure 5) depicts the opposite of the previous, in which a multiple drifting data streams merge together. As before, the stationary model (top panel) cannot accurately represent the component structure or the predictive variance, while the nonstationary approach

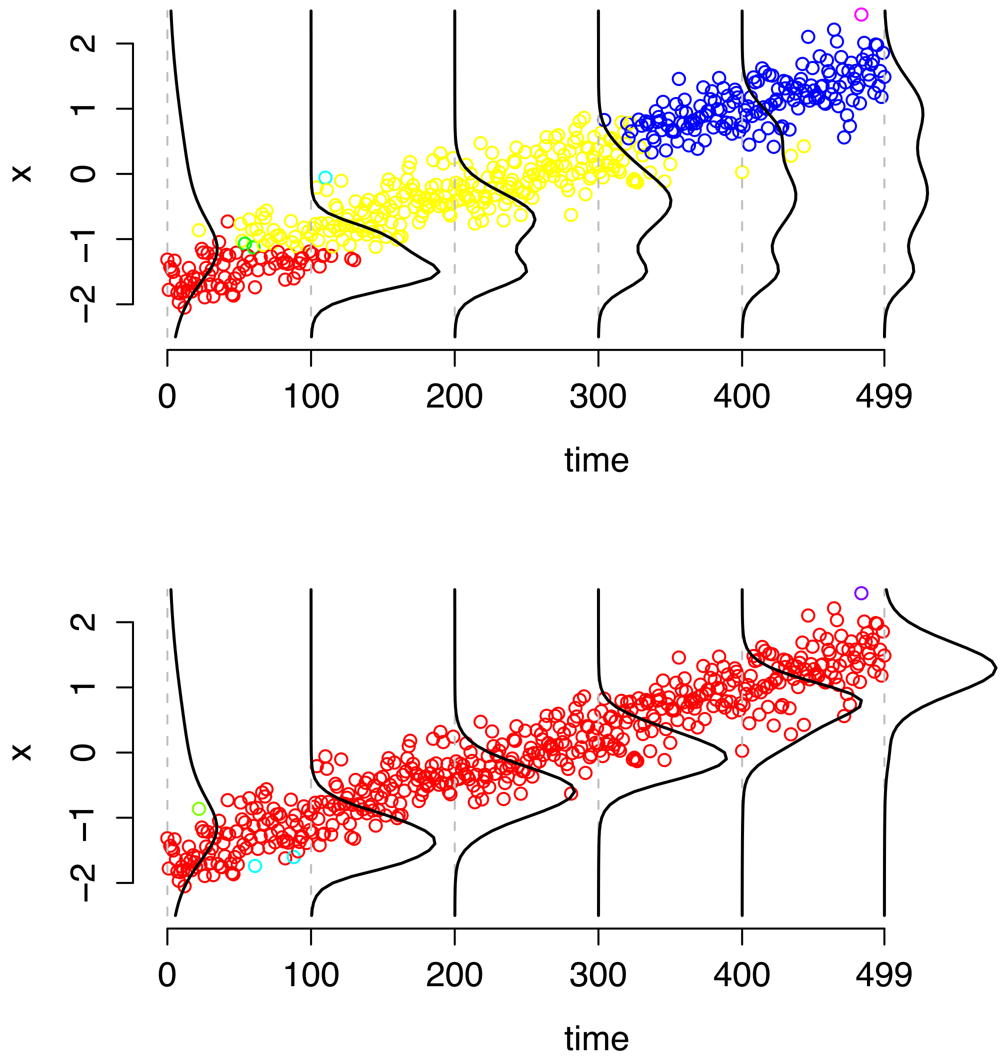


Figure 3: A drifting data stream. Each data is color-coded by its maximum probability label as evaluated at its time of arrival. Predictive distributions are shown vertically, as evaluated at regular time intervals. Under the assumption of stationarity (top panel), the variance of the predictive distribution increases to accommodate the trend in the data, while the predictive distribution of the nonstationary mixture (bottom panel) tracks the trend in the data without an increase in variance.

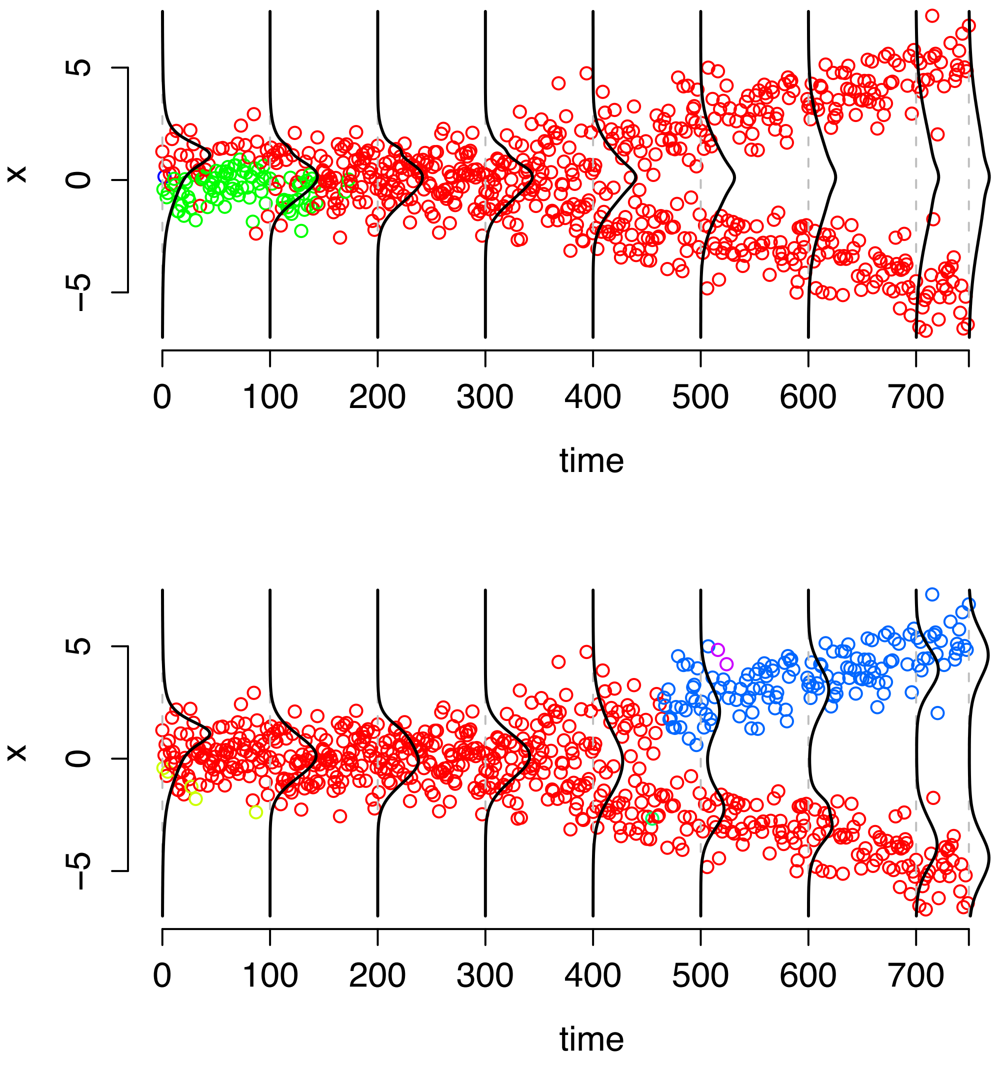


Figure 4: A single data stream splitting into two.

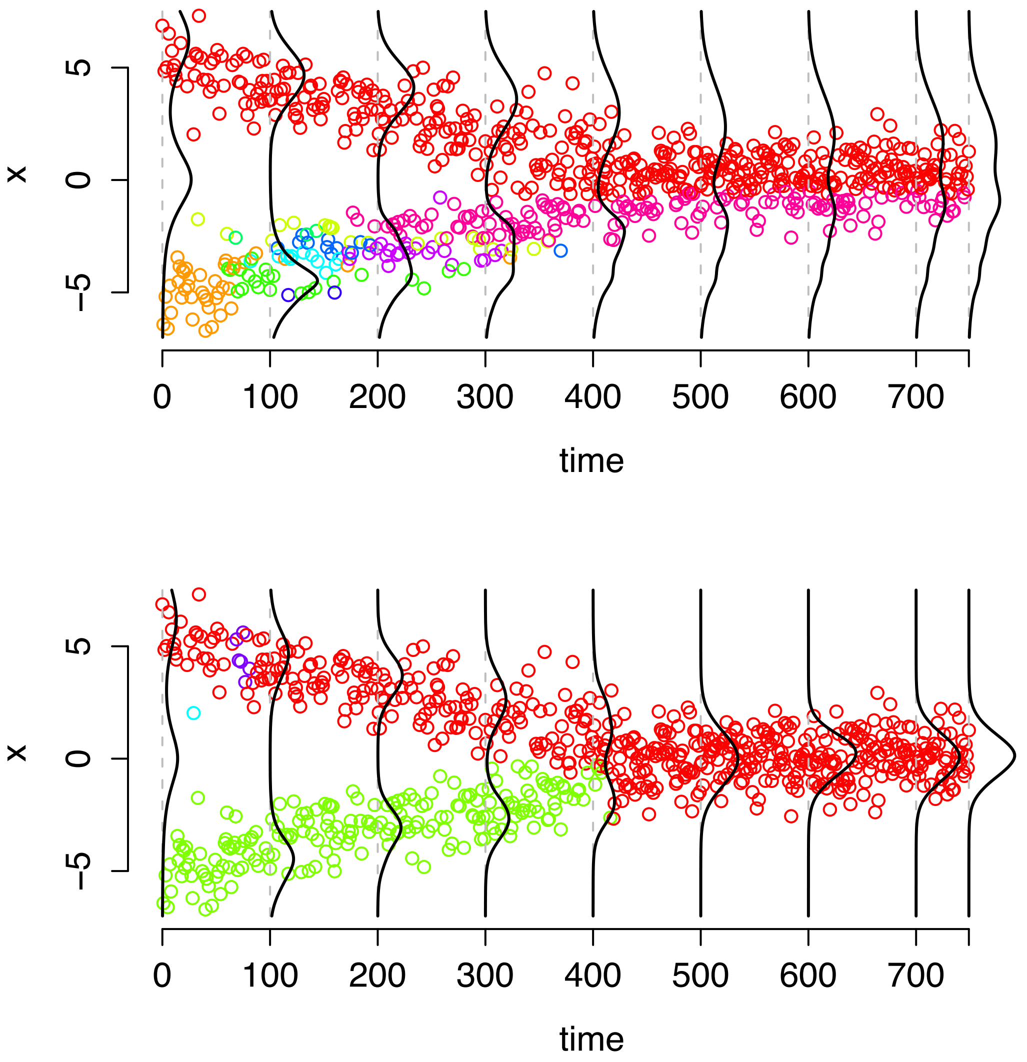


Figure 5: Two data streams merging into one.

(lower panel) tracks the convergence of the two data streams while maintaining a tight bound on predictive variance.

4.3 Iris Data

Next we consider a well studied horticultural dataset attributed to Fisher [5] consisting of 150 measurements of sepal length, sepal width, petal length, and petal width, for a collection of irises. The irises can be classified into one of three types, *Iris Setosa*, *Iris Versicolour*, and *Iris Virginica*, and each type appears with equal frequency. Figure 6 depicts the triplet scatterplots of the data. From Figure 6 it is clear that though the Iris Setosa samples might be easily distinguished from the other other two, distinguishing between Iris Versicolour and Iris Virginica on the basis of the collected data will be challenging. After using PL to fit the DPM to these data, we evaluated the cluster membership probabilities for each observation, for each label generated during the inference process. The probability of label ℓ for observation x_t can be evaluated as

$$\pi_\ell^t \propto \frac{1}{N}\sum_i \sum_{j:l_j=\ell} \frac{n_j}{\alpha_0^i + T} St_{\nu+n_j}\left(x_t|\mu_j, \frac{1+\gamma n_j+\gamma}{(1+\gamma n_j)(\nu+n_j)}\Sigma_j\right) \tag{9}$$

Figure 7 shows the probability of each label for each observation. The model essentially succeeds in distinguishing the Setosa samples, but has difficulty registering the difference between Versicolour and Virginica. Inspection of the empirical density estimates for of each of the four dimensions of the observed data, separated into the distinct Iris subtypes, reveals the distributional similarities between Versicolour and Virginica (Figure 8). Based on the marginal densities, it is clear that only through training will the mixture model be able to represent the subtle differences between Versicolour and Virginica.[3] We constructed a trained particle cloud as described above by conducting three separate analyses using 40 of the 50 samples for each iris subtype. Figure 9 shows the out-of-sample label prediction for the remaining 10 samples of each type using the trained particle cloud. Thus even modest training with a few samples of each subtype proves to be sufficient for capturing the subtle differences between Versicolour and Virginica. Figure 10 contains predicted label probabilities for all 150 samples, and can be compared to Figure 7 to further demonstrate the utility of a training process.

5 Discussion

We have presented several new approaches for Bayesian model based clustering made possible through particle learning techniques. Although all examples considered were based on a multivariate normal mixtures, the ideas pertaining to labeling and adaptive classification pertain to mixtures of any basis distribution. Of particular interest will be the application of these methods for a mixture-of-mixtures model as described in Merl and West [11]. In that model, the basis distributions of a nonparametric mixture model are themselves nonparametric mixtures of normals. Since a mixture of normals can approximate any continuous distribution to an arbitrary

[3] In fact, previous studies of these data have regarded the latter two subtypes as *not linearly separable*.

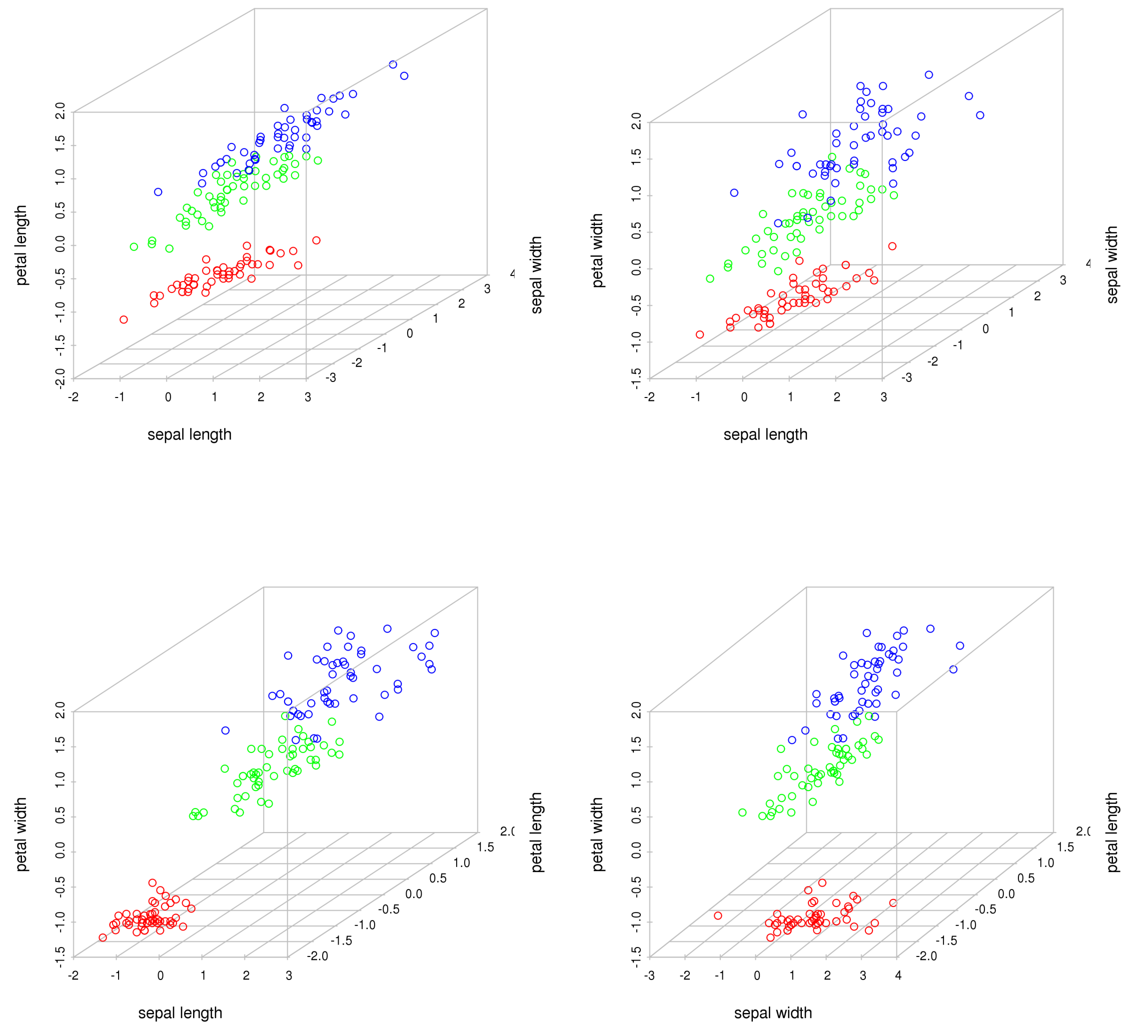


Figure 6: Triplet scatterplots for the iris data. Iris Setosa samples are shown in red, Iris Versicolor in green, and Iris Virginica in blue.

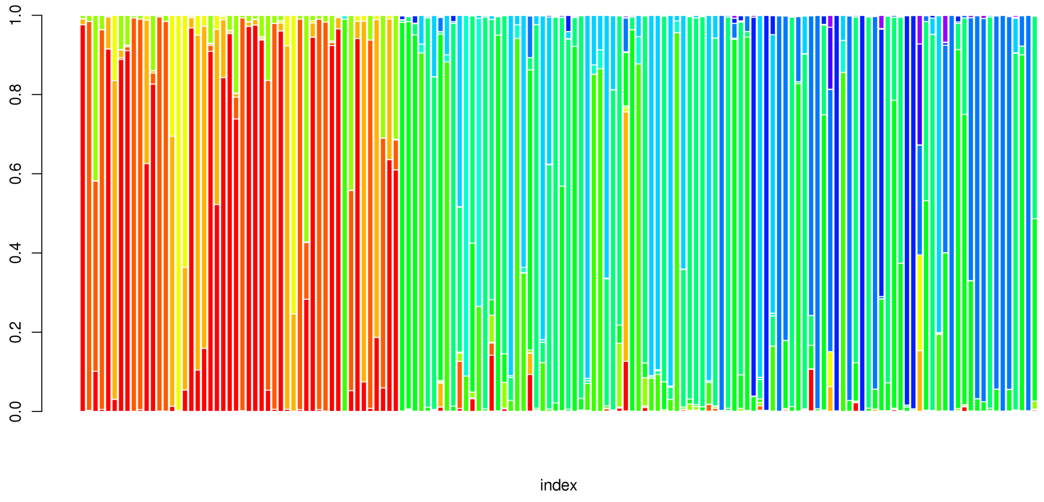


Figure 7: Posterior probabilities associated with each generated label for the observed iris data.

degree of accuracy, the mixture-of-mixtures approach lends itself to classification problems where the distributions of observations comprising different clusters may be highly non-Gaussian. The methods described here can also be made to accommodate a variety of non-continuous data types for which conjugate models are available, such as the multinomial/Dirichlet model for categorical data.

There is an interesting connection to be made between particle learning classifiers and the standard ensemble classifiers of the machine learning literature. In a sense, the former is an instance of the latter, which each particle representing a single learner amongst an ensemble/cloud of similar learners. Classification, as described here, is based on a consensus amongst learners. The key difference between the two has to do with the ease with which the classification rule can be updated in light of new data. In principle, the process of continuing training is straightforward, and a successful demonstration of which would constitute a significant advance in the field. Above we suggested the application of the methods described here to provide support for other classifiers by detecting significant changes to the underlying distribution features and facilitating their retraining by producing labels that capture the effects of these shifts. In fact, it may be the case that this capacity for constant training and updating will cause the framework described here to be outrightly preferred to other classifiers in situations where the stability of the classification rules are in question.

A Updating α_0, μ_0, and Σ_0

Inference may be conducted on hyperparameters α_0, μ_0, and Σ_0 as follows. If the standard α_0 $Gamma(e,f)$ $(E(\alpha_0) = e/f)$ prior distribution is assumed, then the data augmentation

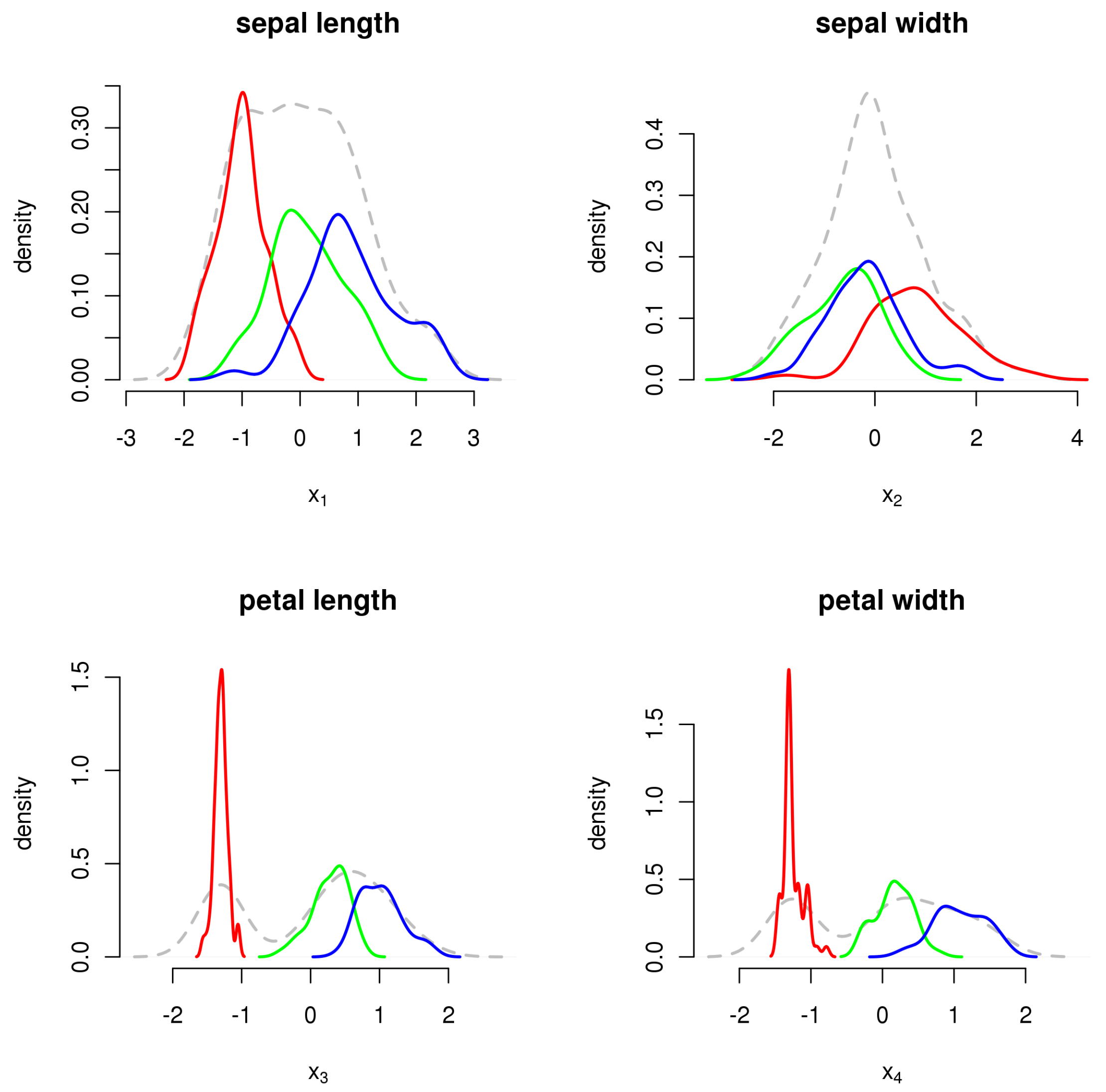


Figure 8: Marginal densities for each dimension of the observed data. Dashed grey density is based on all observations. Densities for Iris Setosa, Iris Versicolour, and Iris Virginica samples are shown in red, green, and blue respectively, scaled in proportional to subtype frequency.

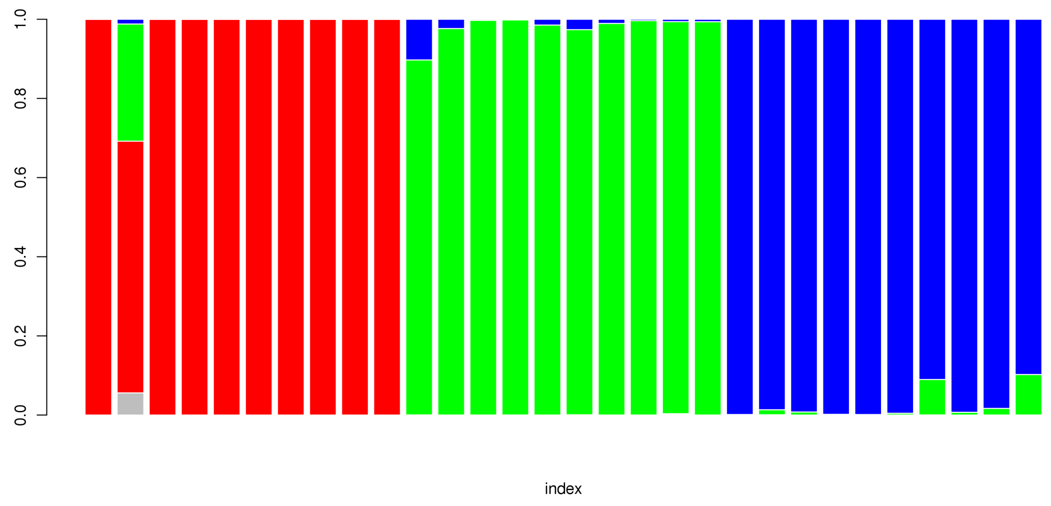


Figure 9: Predicted label probabilities for 10 samples each of Iris Setosa, Iris Versicolour, and Iris Virginica held out during training. Grey indicates probability of a yet-unseen subtype, while red, green, and blue indicate probabilities of the Setosa-, Versicolour-, and Virginica-associated labels respectively.

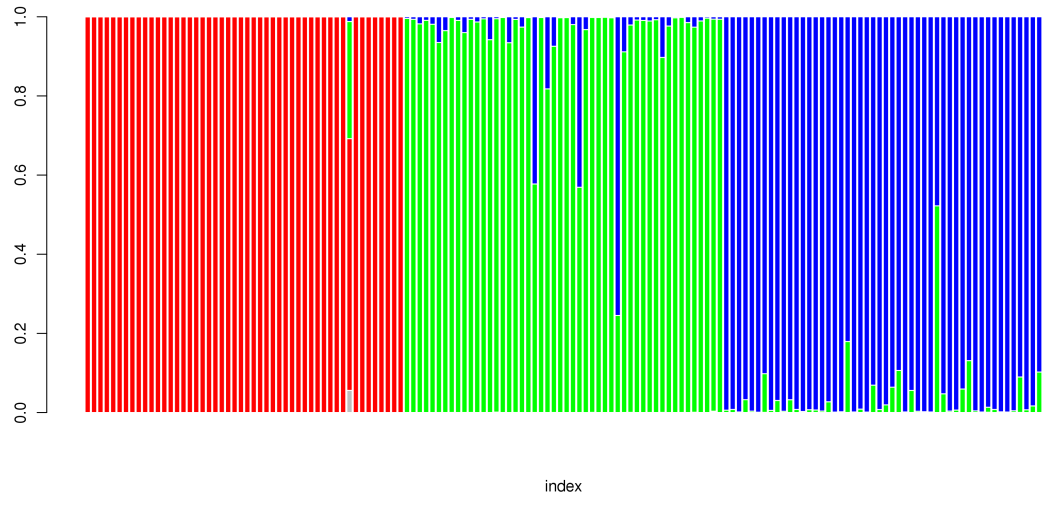


Figure 10: Predicted label probabilities for all iris samples using trained cloud.

scheme of Escobar and West [4] can be used to generate samples from the posterior distribution of α_0. Under this scheme, an auxiliary variable ζ is sampled from a $Beta(\alpha_0 + 1, t)$ distribution, where α_0 is the current value of the parameter, and t is the current number of observations. Then conditional on ζ, a new value of α_0 is sampled from a mixture of $Gamma$ distributions given by:

$$pGamma(e + n^*, f - log(\zeta)) + (1 - p)Gamma(e + n^* - 1, f - log(\zeta))$$

where

$$p = \frac{e + n^* - 1}{t(f - \log(\zeta)) + e + n^* - 1}.$$

As before, n^* denotes the current number of mixture components. This sampling procedure is repeated for each particle, using each particle's α_0 and n^* values.

Samples from the posterior distribution of μ_0 and Σ_0 can be obtained using standard conjugacy results. Assuming independent prior distributions of

$$\begin{aligned} \mu_0 &\sim N(m_0, \Phi_0) \\ \Sigma_0 &\sim Wishart(\nu_0, \Lambda_0/\nu_0) \ (E(\Sigma_0) = \Lambda_0) \end{aligned}$$

then the sampling procedure is the following. First, samples of (μ_j, Σ_j) must be produced for $j = 1 \ldots n^*$ (the mean and covariance matrices for each normal component in the mixture model are not sampled directly in the algorithm described since it is possible to integrate over their Normal-Inverse-Wishart posterior distribution). These posterior distributions are given by

$$(\mu_j, \Sigma_j) \sim NIW_{\nu + n_j}(\mu'_j, \frac{\gamma}{1 + \gamma n_j}, \Sigma'_j)$$

where

$$\begin{aligned} \mu'_j &= \frac{\mu_0 + \gamma n_j \bar{x}_j}{1 + \gamma n_j} \\ \Sigma'_j &= \Sigma_0 + \sum_{i:k_i = j} (x_i - \bar{x}_j)'(x_i - \bar{x}_j) + \frac{n_j}{1 + \gamma n_j}(\mu_0 - \bar{x}_j)'(\mu_0 - \bar{x}_j) \end{aligned}$$

Using these values, the posterior distributions for μ_0 and Σ_o can be sampled as $\mu_0 \sim N(m', \Phi')$ and $\Sigma_0 \sim Wishart(\nu', \Lambda')$ where

$$\begin{aligned} \Phi' &= \left(\Phi_0^{-1} + \sum_{j=1}^{n^*} (\gamma \Sigma_j)^{-1} \right)^{-1} \\ m' &= \left(m_0 \Phi_0^{-1} + \sum j = 1^{n^*} \mu_j (\gamma \Sigma_j)^{-1} \right) \Phi' \end{aligned}$$

and

$$\begin{aligned} \nu' &= \nu_0 + n^*(\nu + d + 1) \\ \Lambda' &= \left(\nu_0 \Lambda_0^{-1} + \nu \sum_{j=1}^{n^*} \Sigma_j^{-1} \right)^{-1} \end{aligned}$$

where d is the dimension of the data. As above, this procedure is repeated for each particle, using each particle's sufficient statistics in order to sample the set of (μ_j, Σ_j), and conditional on those values, (μ_0, Σ_0).

B Dahl's Sequentially Allocated Split/Merge Steps

A common problem in iterative methods for sampling the latent allocation variables used in mixture models is that when the allocation variables are sample one at a time, it can be difficult for the algorithm to effect big changes to the underlying mixture component structure. For example, in Figure 5, under a one-at-a-time sampling approach, in order to recognize the merger of the two mixture components, each observation currently assigned to one component would have to be re-assigned to the other component. Though this would eventually happen under a one-at-a-time approach, the merger process can be greatly facilitated by considering a single MCMC step that proposes the complete merge of the two components into a single component. This can be seen as a type of Metropolis step, proposing a joint update of many parameters rather than the usual update of a single parameter. Jain and Neal [9] described one such approach for generating random joint update steps. Dahl [3] presented an alternative that seems preferable for the application described here for reasons that will become apparent. The basic idea of the sequential split/merge step is as follows.

1. Randomly (and uniformly) choose two observations, x_t and $x_{t'}$.

2. If x_t and $x_{t'}$ have been allocated to the same mixture component, propose to split that component into two components, one founded by observation x_t and one founded by observation $x_{t'}$. If x_t and $x_{t'}$ have been allocated to different components, propose to merge the two components into a single component.

3. Evaluate the acceptance probability of the split or merge, and accept or reject the move accordingly.

The key difference between Dahl and Jain and Neal lies in the way the reassignment of allocation variables occurs during a split. Under Dahl's approach, each observation currently allocated to the common component is reassigned to one of the new components. Initially one component contains only x_t and the other $x_{t'}$. The proposed split is determined by cycling through the observations currently assigned to the common component, successively adding to the component founded by x_t or the component founded by $x_{t'}$. This reallocation step is accomplished by sampling from the usual Polya urn predictive distribution, considering only these two components.

Thus, the interpretation of this reallocation is simply, *how would the observations assigned to this component have changed if x_t and $x_{t'}$ had been observed first ? (instead of whatever observation founded the component).*

The Metropolis Hastings acceptance probability of the move is evaluated as $\frac{p(k^t_{new}|x^t)q(k^t_{old}|k^t_{new})}{p(k^t_{old}|x^t)q(k^t_{new}|k^t_{old})}$, where k^t denotes the set of allocation variables up to time t and q is the proposal probability. The posterior probability of the a set of allocation variables can be evaluated directly by combining the likelihood of the data under the proposed partition (which follows the usual Polya urn structure) and the prior probability of the partition induced by the Dirichlet process. The probability of the proposal is either 1 (for a merge move) or it is obtained as the product of the successive reassignment probabilities used to generate the split in step 2 above.

For the application presented here, there are two technical considerations. The first has to do with the situation in which the decay factor λ is less than 1. In this case, reassignment is performed using only the most recent $1/(1-\lambda)$ observations currently in the sliding window. The second technical detail has to do with how labels are preserved after a split or merge is accepted. In the examples presented here, we have adopted a relabeling scheme as follows. When a split is accepted, the larger of the two new components retains the original label, and the other component is labeled by t or t' as appropriate. When a merge is accepted, the new component retains the label of the earlier of the two components it is comprised of. These relabeling rules were chosen somewhat arbitrarily, but they have proven useful in practice. Other relabeling schemes are possible.

C A very fast sequential approximation to the full model

Although the countably infinite mixture model induced by the Dirichlet process prior on the mixing distribution provides a very popular framework for methodological work on model-based clustering, practitioners have become increasingly frustrated with the scalability of the associated inference algorithms. There exist several precedents for adopting adopting fast approximate inference schemes for mixture models. Han et al [6] discuss one such approach in the context of online visual tracking systems. Their approach involves sequentially recalculating the modes of a kernel density estimate upon arrival of a new datum, and consolidating mixture components so that a single component is associated with each mode. Their approach is appealing for its sequential (and linear time) nature, but it lacks the ability to accommodate arrivals from nonstationary mixture distributions. Additionally, since their algorithm is at its core a kernel density estimation, there is limited opportunity for quantifying uncertainty regarding the *outlierliness* of an observation (our preference for formal probability models which regard the number of components as random quantities to be estimated from the data stems from their abilities to produce such statements). More recently, Daume III [8] presented a search heuristic for deriving approximate MAP component assignments for a Dirichlet process mixture model. This approach appears to perform on par with standard Gibbs samplers in a fraction of the time, though the authors comment that the algorithm is still not competitive with variational methods in terms of speed and accuracy. The approach is interesting, but is incapable of accommodating streaming data due to its reliance on an upper bound on the marginal likelihood of the remaining data.

Sequentially learned nonparametric mixture models have many desirable properties in terms of inference, viz., accommodating nonstationarity while simultaneously facilitating clustering and anomaly detection. However, in some application spaces it may computationally prohibitive to maintain an entire particle cloud *and* achieve the desired between-observation latency in order that the inference algorithm can remain in lock-step with the arrival of the data. In order to accommodate such applications, we now describe a much faster, approximate procedure *inspired by* the full particle learning algorithm as previously described in Section 2.

C.1 Method

The key ideas of this approach are the following:

- Maintain only one instantiation of the mixture model (as opposed to a cloud of particles each containing a distinct instantiation of the mixture). This instantiation will represent, in some sense, the expected mixture distribution at time t conditional on the model state at time $t-1$. This distribution is similar to the distribution that would be obtained in the full particle learning approach by averaging the distributions of each particle, i.e.

$$\frac{1}{N}\sum_{i=1}^{N} p(x_t|\{s_j\}_{j=1}^{n_i^*}, \alpha, \mu_0, \Sigma_0)$$

- Sequentially update this distribution in order to maximize the predictive probability of the next observation.

- During update steps, decay the contribution of previous observations towards the sufficient statistics of each mixture component by a factor of λ. This will induce a sort of sliding window model, in which the effective total number of observations is $\frac{1}{1-\lambda}$. Note that this step is optional, though its inclusion is required for handling arbitrarily long-lasting data streams.

- Between *deterministic* update steps, perform *random* split/merge steps to as described in Appendix B and [3] using the most recent $\frac{1}{1-\lambda}$ observations.

At its core, this appears to describe something similar to one pass of an EM algorithm. Normally EM would involve *many* passes through the data, repeatedly updating the configuration variables for each observation until convergence is reached according to some criterion. One pass of EM is certainly not guaranteed to produce any sort of reliable estimate of the MAP cluster assignment. However, the inclusion of the split/merge steps proves to be key. Upon arrival of a new observation, the fundamental question is, *does this observation fall into one of the clusters observed thus far, or does it constitute the first arrival from a new cluster, or does it change our opinions about what we are currently calling clusters?* The first part of the question is answered by the prediction rule for the Polya urn representation of the Dirichlet process mixture. The second part is answered through splitting and merging existing clusters.

An informal argument in support of *why* such an approach should be expected to work is as follows. Let $c^t = [c_1\ c_2\ c_3\ \ldots\ c_t]$ denote the component/cluster assignment variables for the first t observations, and let $\widehat{c^t}$ denote the maximum a posteriori estimate of c^t based on the data up to time t. In other words,

$$\widehat{c^t} = \arg\max_{c^t} p(c^t | x^t, \alpha, \mu_0, \Sigma_0)$$

where $x^t = [x_1\ x_2\ x_3\ \ldots\ x_t]$. Upon arrival of the next observation x_{t+1}, one of two things can happen: either $\widehat{c^{t+1}} = [\widehat{c^t}\ c_{t+1}]$ or it does not. Another way of saying this, by analogy, is that either x_{t+1} is the straw that breaks the camel's back or it is not. Consider the following factorization of the posterior distribution of c^{t+1}:

$$p(c^{t+1}|x^{t+1}, \alpha, \mu_0, \Sigma_0) = p(c_{t+1}|c^t, x^{t+1}, \alpha, \mu_0, \Sigma_0) p(c^t|x^{t+1}, \alpha, \mu_0, \Sigma_0).$$

If the arrival of x_{t+1} does not change our way of thinking about the cluster assignments of the first t observations, then the posterior distribution of c^{t+1} will be maximized by assigning x_t to the component that maximizes $p(c_{t+1}|c^t, x^{t+1}, \alpha, \mu_0, \Sigma_0)$, as is accomplished by the EM step described above. If the arrival of x_{t+1} makes it such that $\widehat{c^{t+1}}$ involves cluster reassignments for some of the first t observations (relative to their assignment in $\widehat{c^t}$), then with high probability that reassignment will be discovered by the random split/merge steps. For example, suppose that data arrives as in Figure 11. Up until $t \approx 250$, $\widehat{c^t}$ will involves assigning all observations into one of two clusters, however at some point it becomes clear that the data (if assumed to come from a stationary mixture) in fact represent a single cluster. In order to recover the MAP estimate, the two existing components must be merged to a single component, as will be accomplished by the split/merge steps.

In full detail, the algorithm is as follows.

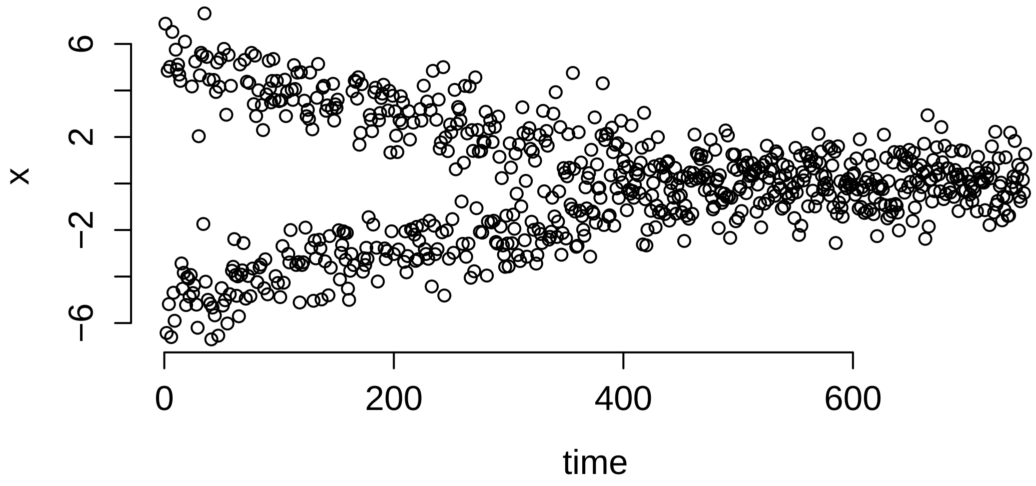
1. After the arrival of x_1, initialize the sufficient statistics for component 1 with the appropriate function of x_0 (i.e. $s_1 = [x_1\ \ x_1'x_1\ \ 1]$). At this point, $c_1 = 1$ and trivially $\widehat{c_1} = [1]$.

2. After the arrival of x_t, sample c_t as

$$c_t = \arg\max_\ell \left\{ \frac{\alpha}{\alpha + t - 1} p(x_t|\mu_0, \Sigma_0)\delta_{\ell = n^*+1}(\ell) + \sum_{i=1}^{n^*} \frac{n_i}{\alpha + t - 1} p(x_t|s_i, \mu_0, \Sigma_0)\delta_{\ell=i}(\ell) \right\}$$

3. If $c_t = n^* + 1$, initialize a new component with $s_{n^*+1} = [x_t\ \ x_t'x_t\ \ 1]$. Otherwise update the sufficient statistics for component c_t using the decay factor λ: $s_{c_t} = s_{c_t}\lambda + [x_t\ \ x_t'x_t\ \ 1]$.

4. Randomly choose i and j from among indices in the range $[t - 1/(1-\lambda) \ldots t]$ (i.e. the most recent $1/(1-\lambda)$ observations). If observations i and j are currently assigned to the same component, propose to split the component into two components, seeded by observations i and j. If i and j are in different components, propose to merge the two components



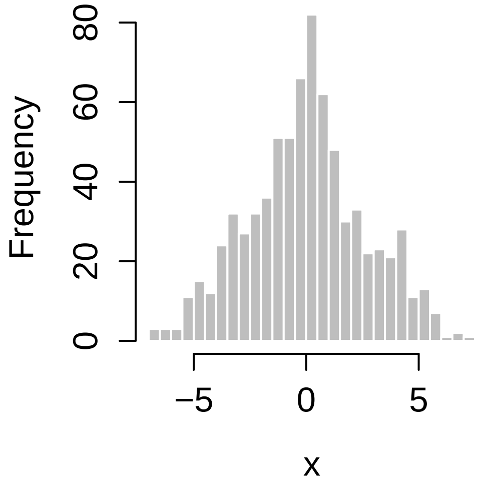


Figure 11: Sample data.

into a single component. The full details for this sampling step appear in [3]. Evaluate the Metropolis-Hastings acceptance probability for this move, and accept the move if the probability is greater than 0.5. Essentially creates a randomized EM step for splitting and merging components.

5. Repeat steps 2–4 for each new observation.

C.2 Demonstration

In this section we use simple simulations to demonstrate the key features of this approach, and to compare the approximate results to those obtained using the full PL algorithm. These simulations are the same as those presented in Section 4 above. In each of the following figures, we show the data, color coded by label, along with the predictive distribution as evaluated at several time points over the course of observation. First, we consider 500 observations from a normal regression model with Gaussian noise. Figure 12 shows the clustering and predictive densities induced under a decay factor of $\lambda = 0.99$. The predictive density successfully tracks the

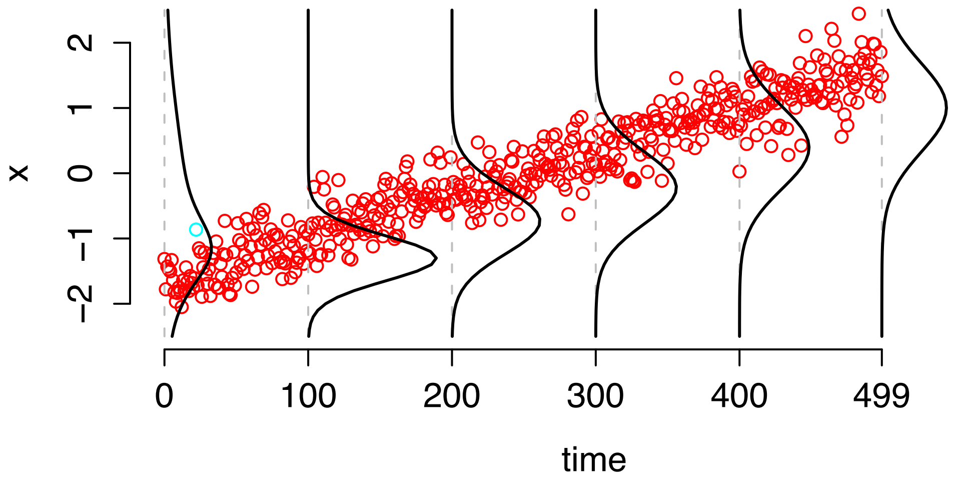


Figure 12: Simulated drifting data stream, color coded by cluster as estimated by nonstationary approximation, with the estimated predictive distributions overlayed.

nonstationary distribution generating the data, and therefore allows recognition of a common cluster assignment for all observations. The next two simulations demonstrate the utility of the split/merge steps of the algorithm. These simulations were constructed to represent situations in which the "true" cluster structure is only revealed in retrospect. In the context of data streams, this amounts to situations in which the data streams can be seen to drift, split, and merge with one another over time. Figure 13 shows the results of the algorithm applied to the pathological merging data shown previously in Figure 11. Using a decay factor of $\lambda = 0.99$ and a single ran-

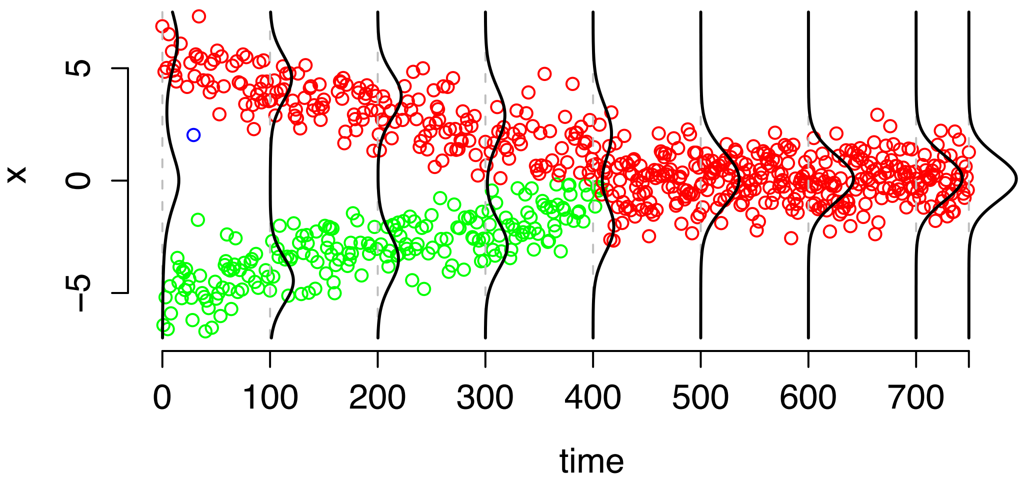


Figure 13: Simulated merging data streams, color coded by cluster as estimated by nonstationary approximation with split/merge steps, with the estimated predictive distributions overlayed.

	approx	(time/obs)	full	(time/obs)
drift data	2.879s	(0.005758s)	242.8s	(0.4857s)
merge data	3.863s	(0.005151s)	341.9s	(0.4559s)
split data	3.960s	(0.005280s)	347.5s	(0.4633s)

Table 1: Comparison of computation times required for simulation studies. The approximate method is approximately two orders of magnitude faster than the full method.

dom split/merge step after each observation, the algorithm initially partitions the observations into two separate streams, before ultimately merging them into a single stream around time 400. Similarly, Figure 14 shows the results of tracking a single data stream that ultimately splits into two diverging streams. Visual comparison of these results to those presented in Section 4 indicate that the approximate method performs as well as the full method. In the case of the splitting data stream, although the split is similarly detected, here the upper substream retains the label of the original stream. This is reversed from that presented in in Section 4, and simply reflects the fact that there is uncertainty as to which of the newly spawned mixture components has larger weight. Compute times required for these analyses, as compared to those required for the full method, are presented in Table 1, and further emphasize the utility of the approximate method. These times are associated with single-threaded versions of each algorithm running on one core of a 2.4 GHz Intel Core 2 processor. The approximate method appears to be nearly two orders of magnitude faster than the full method.

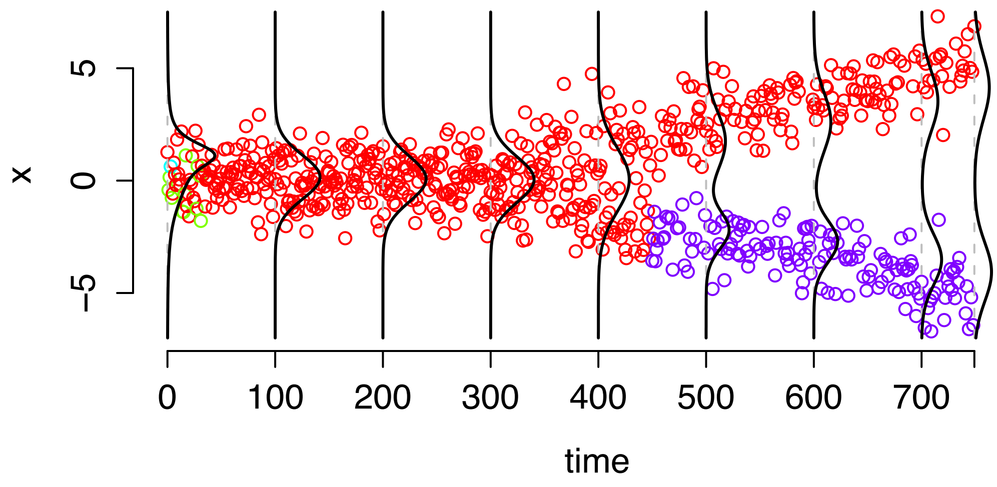


Figure 14: Simulated splitting data streams, color coded by cluster as estimated by nonstationary approximation with split/merge steps, with the estimated predictive distributions overlayed.

References

[1] C. Carvalho, M. Johannes, H. Lopes, and N. Polson. Particle learning and smoothing. *Duke DSS Discussion Paper*, 2009.

[2] C. Carvalho, H. Lopes, N. Polson, and M. Taddy. Particle learning for general mixtures. *Duke DSS Discussion Paper*, 2009.

[3] D. Dahl. Sequentially-allocated merge-split sampler for conjugate and nonconjugate Dirichlet process mixture models. *TAMU Dept of Statistics Technical Report*, 2005.

[4] M. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588, 1995.

[5] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7:179–188, 1936.

[6] B. Han, D. Comaniciu, Y. Zhu, and L. Davis. Sequential kernel density approximation and its application to real-time visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1186–1197, 2008.

[7] G. Hendeby, J. Hol, R. Karlsson, and F. Gustafsson. A graphics processing unit implementation of the particle filter. Technical report, Department of Electrical Engineering, LinkÃűping University, 2007.

[8] H. Daumé III. Fast search for Dirichlet process mixture models. *Conference on AI and Statistics*, 2007.

[9] S. Jain and R. Neal. A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13:158–182, 2000.

[10] S. Maskell, B. Alun-Jones, and M. Macleod. A single instruction multiple data particle filter. In *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, pages 51–54, 2006.

[11] D. Merl and M. West. Mixtures of mixtures for model based clustering. *Duke DSS Discussion Paper*, 2009.

[12] M. Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 62:795–809, 2000.

[13] M. West and J. Harrison. *Bayesian forecasting and dynamic models*. Springer-Verlag, 1997.